Los Alamos NATIONAL LABORATORY

CORE: A Software Component for Conservative Remapping

Martin Staley, mstaley@lanl.gov
Mikhail Shashkov, shashkov@lanl.gov

Conservative Remapper, or CORE, is a C++ language software library for remapping quantities defined on unstructured two dimensional grids. CORE contains implementations of two remapping algorithms: a new, efficient "swept region" algorithm, and a more traditional algorithm based on the computation of cell intersections. Grids may be Cartesian or cylindrical, and cells may have three or more vertices, with no upper limit. CORE can run in serial and in parallel, but in order to achieve wide applicability, it uses no particular parallel communication library. Instead, it achieves parallel communication through strategically placed, user-defined callbacks. Users can also redefine different parts of the remapping process by providing callbacks. CORE allows for different data types, e.g. single, double, and quad precision, through its use of C++ templates. Using CORE is simple, and requires no configuration scripts or makefiles.

Swept-Region Algorithm

The swept-region remapping algorithm [1] achieves its goal by performing three stages, which we will call density reconstruction, mass exchange, and mass repair. This algorithm performs well only if the new grid is a small perturbation of the old grid.

For the **density reconstruction** stage, consider that the mean cell densities obey some underlying, theoretical density function ρ. We don't know what ρ is, but if we assume it is piecewise linear—one piece per cell—then we can use our discrete mean densities to reconstruct a reasonable candidate. For each cell, we consider a linear density function that achieves the cell's mean density at the cell's center of volume. Then, we use values of mean density in nearby cells to compute a reasonable gradient for the function. With a point and a gradient, we have our linear density function for the cell.

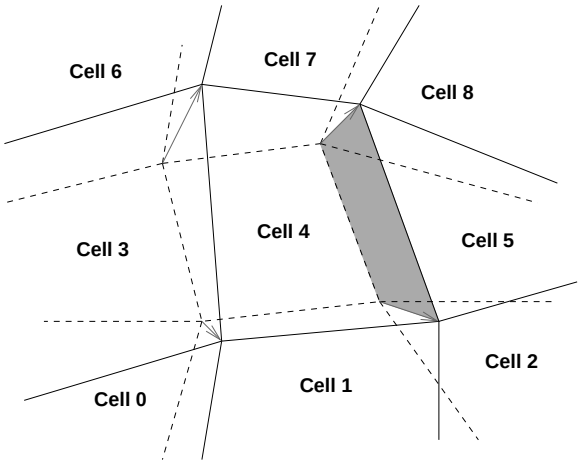

Figure 1. *Swept-region remapping. We determine which of its left and right cells an edge sweeps into the most when it moves from the old grid to the new grid.*

For the **mass exchange** stage, consider that each edge has two adjacent cells. For each edge, we identify which cell the edge "moves into" more when we go from the old grid to the new grid. We then compute the integral of that cell's reconstructed density over the region swept by the edge as it moves from its old position to its new position. The resulting mass is removed from that cell and added to the other one.

Consider Cell 4 in Figure 1. Its vertices move from their positions on the old grid (dotted lines) to their positions on the new grid (solid lines) as shown by the arrows. Swept-region remapping examines each of the grid's edges. Consider Cell 4's right edge, that is, the edge between Cell 4 and Cell 5. The algorithm considers which of those two cells the edge tends to sweep into when we go from the old grid to the new grid. In this case, the answer is Cell 5. So, the algorithm computes a mass term by integrating Cell 5's reconstructed density function over the edge's swept region: the shaded region in the figure.

The fact that the swept region also intersects with cells 2 and 8 is ignored by this algorithm. Rather, the region "mostly" intersects with Cell

5, and therefore, only Cell 5's reconstructed density function is considered. The resulting mass is removed from Cell 5 and added to Cell 4. (In general, the cell that an edge sweeps *into* has its mass reduced due to that edge's movement, while the cell that an edge sweeps *out of* has its mass increased due to that edge's movement.) The same procedure is applied to each edge of the grid.

Finally, for the **mass repair** stage, we first recognize that the mass exchange stage involved inexact integration of the overall density function ρ, in the sense that the mass exchange term was obtained by computing the integral over the swept region of the reconstructed density function on one particular cell, even though the swept region may have intersected other cells as well. This made the algorithm efficient, because we did not need to compute exact intersections of new cells with old cells. However, as a result of inexact integration, new masses in individual cells can conceivably violate appropriate bounds (for example, a mass might be negative). The repair stage fixes out-of-bounds masses while conserving total mass.

Exact-Intersection Algorithm

The exact-intersection remapping algorithm [2] begins with the same **density reconstruction** stage used in the swept-region algorithm. In this stage, a reasonable piecewise linear density function is computed.

However, the **mass exchange** stage is now quite different. Consider Cell 4 in Figure 2, which moves from its position on the old grid (dotted lines) to its position on the new grid (solid lines) as shown by the arrows. (You can see that this is the same grid we showed in Figure 1.) Exact-intersection remapping examines each of the grid's cells. Cell 4 on the new grid intersects with cells 1, 2, 4, 5, 7, and 8 on the old grid, as shown by the shaded regions in the figure. So, the algorithm computes Cell 4's new mass by summing the integral of Cell 1's reconstructed density function over Cell 4's intersection with old Cell 1, the integral of Cell 2's reconstructed density function over Cell 4's intersection with old Cell 2, etc.

One might assume that the **mass repair** stage is no longer necessary when we use exact-intersection remapping. After all, its purpose in the swept-region algorithm is to repair out-of-bounds masses that arise because swept regions generally intersect not only with cells the edges tend to move into the most, but with other nearby cells as well, leading to inexact integration of the reconstructed density function. However, if we allow for the possibility that users will define mass bounds in non-standard ways, then in principle, either remapping algorithm can produce out-of-bounds masses. We therefore retain the mass repair stage here.

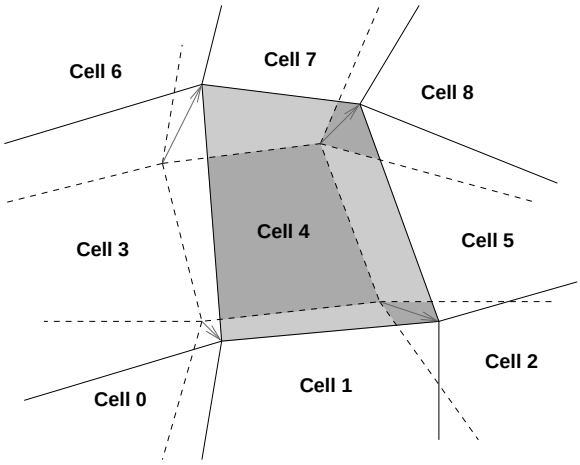

Figure 2. *Exact-intersection remapping. For each cell on the new grid, we determine which cells on the old grid it intersects with.*

The exact-intersection algorithm does *not* require that the new grid is a small perturbation of the old grid.

Acknowledgements

Funded by the ASCI program in the Department of Energy under contracts W-7405-ENG-36. Los Alamos Report LA-UR-04-1319.

References

[1] M. Kucharik, M. Shashkov, and B. Wendroff, *An Efficient Linearity-and-Bound-Preserving Remapping Method*. J. Comp. Phys., 188 (2003) pp. 462–471.

[2] L. Margolin and M. Shashkov, *Second-Order Sign-Preserving Conservative Interpolation (Remapping) on General Grids*, J. Comp. Phys., 184 (2003) pp. 266–298.